Classification_DecisionTrees

October 20, 2025

[1]:

```
# Import necessary libraries
import pandas as pd  # For data manipulation
from sklearn.model_selection import train_test_split  # For splitting data
from sklearn.tree import DecisionTreeClassifier  # For Decision Tree model
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,␣
 ↪recall_score, classification_report  # For evaluation
```

[2]:

```
# Load the Balance Scale dataset directly from UCI repository
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/balance-scale/
 ↪balance-scale.data'

# The dataset has no header, so we specify column names
col_names = ['Class', 'Left-Weight', 'Left-Distance', 'Right-Weight',␣
 ↪'Right-Distance']
data = pd.read_csv(url, header=None, names=col_names)

# Display the first few rows to check the data
print(data.head())
```

```
  Class  Left-Weight  Left-Distance  Right-Weight  Right-Distance
0     B            1              1             1               1
1     R            1              1             1               2
2     R            1              1             1               3
3     R            1              1             1               4
4     R            1              1             1               5
```

[3]:

```
# Separate features (X) and target (y)
X = data[['Left-Weight', 'Left-Distance', 'Right-Weight', 'Right-Distance']]
y = data['Class']
```

[4]:

```
# Split the data: 70% training, 30% testing
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y)
```

[5]:

```
# Create and train the Decision Tree using Gini Index
clf_gini = DecisionTreeClassifier(criterion='gini', random_state=42)
clf_gini.fit(X_train, y_train)
```

```
[5]: DecisionTreeClassifier(random_state=42)
```

```
[6]: # Create and train the Decision Tree using Entropy
     clf_entropy = DecisionTreeClassifier(criterion='entropy', random_state=42)
     clf_entropy.fit(X_train, y_train)
```

```
[6]: DecisionTreeClassifier(criterion='entropy', random_state=42)
```

```
[7]: # Predict on the test set using both models
     y_pred_gini = clf_gini.predict(X_test)
     y_pred_entropy = clf_entropy.predict(X_test)
```

```
[8]: # Confusion Matrix
     print('Confusion Matrix (Gini):')
     print(confusion_matrix(y_test, y_pred_gini))

     # Accuracy
     print('Accuracy (Gini):', accuracy_score(y_test, y_pred_gini))

     # Precision and Recall (macro average for multiclass)
     print('Precision (Gini):', precision_score(y_test, y_pred_gini,␣
       ↪average='macro'))
     print('Recall (Gini):', recall_score(y_test, y_pred_gini, average='macro'))
```

```
Confusion Matrix (Gini):
[[ 2  6  7]
 [14 73  0]
 [13  8 65]]
Accuracy (Gini): 0.7446808510638298
Precision (Gini): 0.6036079182630907
Recall (Gini): 0.5760759155306068
```

```
[9]: # Confusion Matrix
     print('Confusion Matrix (Entropy):')
     print(confusion_matrix(y_test, y_pred_entropy))

     # Accuracy
     print('Accuracy (Entropy):', accuracy_score(y_test, y_pred_entropy))

     # Precision and Recall (macro average for multiclass)
     print('Precision (Entropy):', precision_score(y_test, y_pred_entropy,␣
       ↪average='macro'))
     print('Recall (Entropy):', recall_score(y_test, y_pred_entropy,␣
       ↪average='macro'))
```

```
Confusion Matrix (Entropy):
[[ 2  6  7]
 [14 72  1]
 [15  3 68]]
Accuracy (Entropy): 0.7553191489361702
Precision (Entropy): 0.61604728667547
Recall (Entropy): 0.5838724048828299
```

[10]:

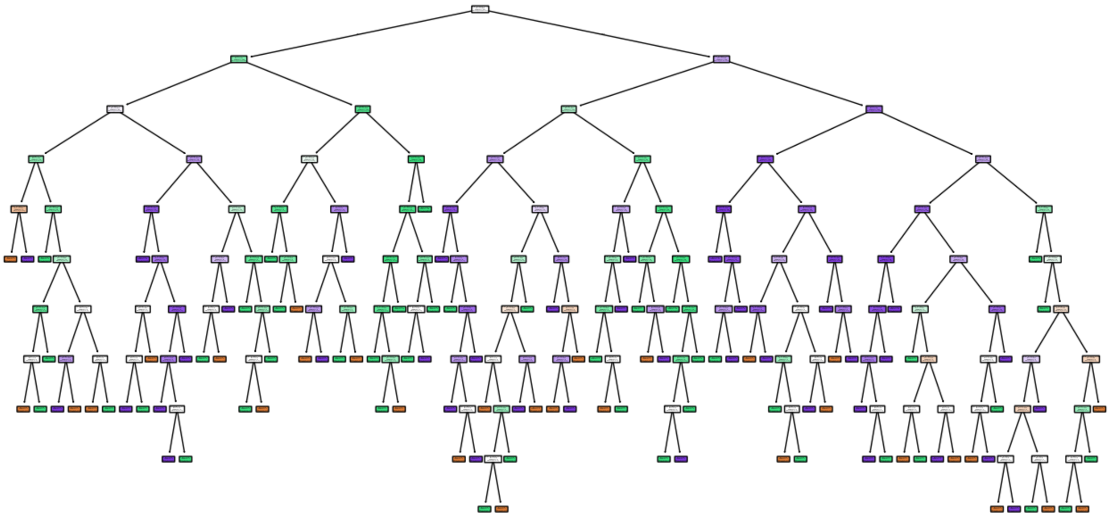
```
# Print detailed classification report for both models
print('Classification Report (Gini):')
print(classification_report(y_test, y_pred_gini))

print('Classification Report (Entropy):')
print(classification_report(y_test, y_pred_entropy))
```

```
Classification Report (Gini):
              precision    recall  f1-score   support

           B       0.07      0.13      0.09        15
           L       0.84      0.84      0.84        87
           R       0.90      0.76      0.82        86

    accuracy                           0.74       188
   macro avg       0.60      0.58      0.58       188
weighted avg       0.81      0.74      0.77       188

Classification Report (Entropy):
              precision    recall  f1-score   support

           B       0.06      0.13      0.09        15
           L       0.89      0.83      0.86        87
           R       0.89      0.79      0.84        86

    accuracy                           0.76       188
   macro avg       0.62      0.58      0.59       188
weighted avg       0.83      0.76      0.79       188
```

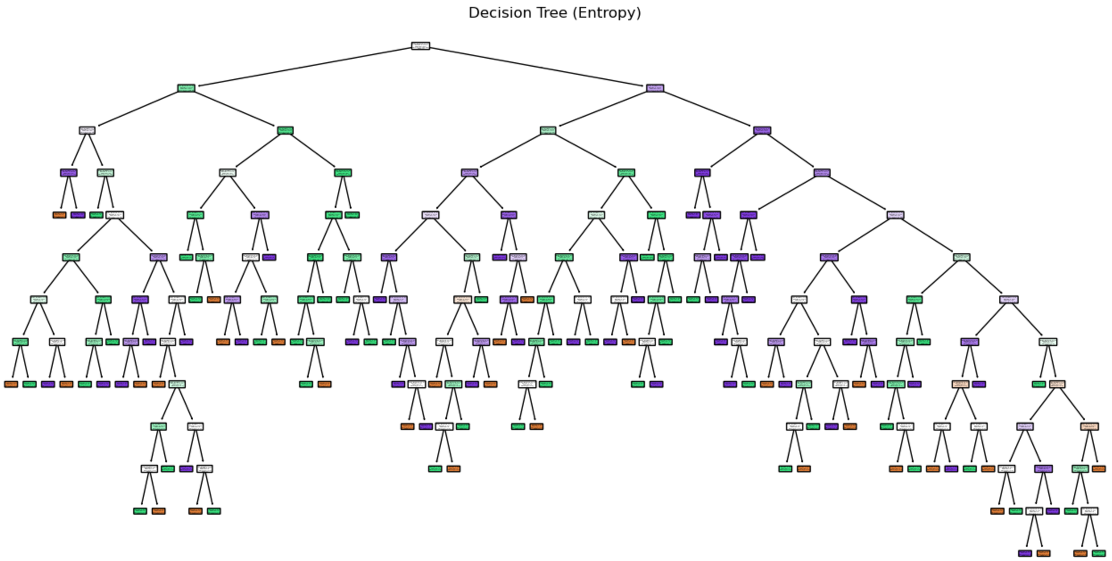
[11]:

```
# Import matplotlib for plotting
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

# Visualize the Decision Tree trained with Gini Index
plt.figure(figsize=(16,8))  # Set the figure size for better readability
plot_tree(clf_gini,
          feature_names=['Left-Weight', 'Left-Distance', 'Right-Weight',␣
 ↪'Right-Distance'],
          class_names=['L', 'B', 'R'],
          filled=True,
          rounded=True)
plt.title("Decision Tree (Gini Index)")
plt.show()
```

Decision Tree (Gini Index)

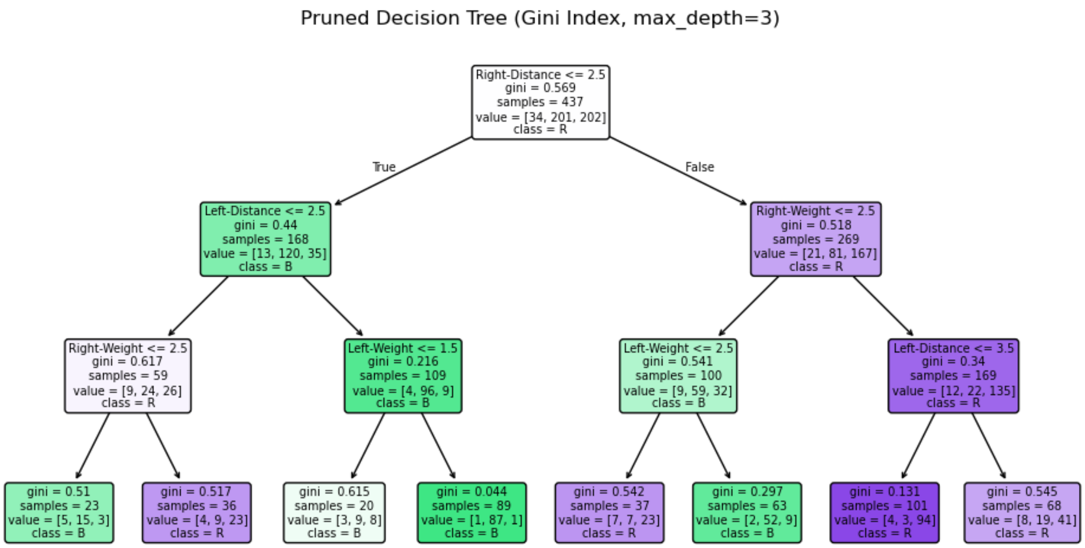


```
[12]: # Visualize the Decision Tree trained with Entropy
plt.figure(figsize=(16,8))
plot_tree(clf_entropy,
          feature_names=['Left-Weight', 'Left-Distance', 'Right-Weight',␣
 ↪'Right-Distance'],
          class_names=['L', 'B', 'R'],
          filled=True,
          rounded=True)
plt.title("Decision Tree (Entropy)")
plt.show()
```



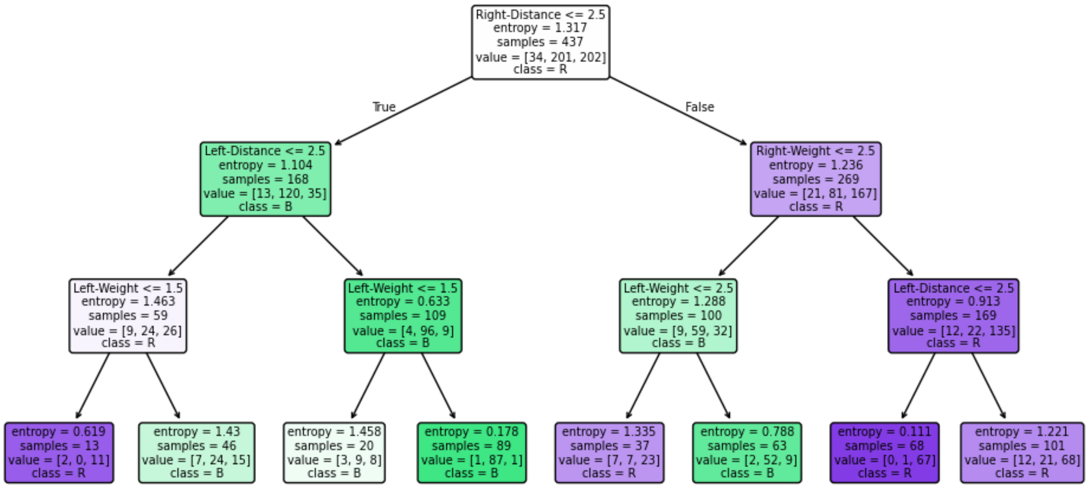
[14]:

```
# Train a pruned Decision Tree with Gini Index
pruned_clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=3,␣
 ↪random_state=42)
pruned_clf_gini.fit(X_train, y_train)  # Fit to training data

# Visualize the pruned tree
plt.figure(figsize=(12, 6))
plot_tree(pruned_clf_gini,
          feature_names=['Left-Weight', 'Left-Distance', 'Right-Weight',␣
 ↪'Right-Distance'],
          class_names=['L', 'B', 'R'],
          filled=True,
          rounded=True)
plt.title("Pruned Decision Tree (Gini Index, max_depth=3)")
plt.show()
```



[17]:
```python
# Train a pruned Decision Tree with Entropy
pruned_clf_entropy = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)
pruned_clf_entropy.fit(X_train, y_train)  # Fit to training data

# Visualize the pruned tree
plt.figure(figsize=(12, 6))
plot_tree(pruned_clf_entropy,
          feature_names=['Left-Weight', 'Left-Distance', 'Right-Weight', 'Right-Distance'],
          class_names=['L', 'B', 'R'],
          filled=True,
          rounded=True)
plt.title("Pruned Decision Tree (Entropy, max_depth=3)")
plt.show()
```

Pruned Decision Tree (Entropy, max_depth=3)



```
[ ]: !jupyter nbconvert --to pdf "Classification_DecisionTrees.ipynb" --output "C:/
     ↪Users/ASUS/Downloads/Classification_Dec
```